NTNU – Trondheim
Norwegian University of
Science and Technology

Title

Vegard Stjerna Lindrup

Submission date: May 2016
Responsible professor: Tor Engebret Onshus
Supervisor:

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Title: Title
Student: Vegard Stjerna Lindrup

Problem description:

Dette legges til i DAIM, og blir derfor fjernet før innlevering.

Responsible professor: Tor Engebret Onshus
Supervisor:

Abstract

Mobile robot platforms etc...

Sammendrag

Mobile robotplatformer kan kjøre rundt og...

Preface

Hva synes jeg om oppgaven? Kjempeartig! Eget acknowledgemet-kapittel?

Acknowledgments

This thesis, and the obtained project results would not have been possible without help from student colleagues, friends and support from the Department of Engineering Cybernetics.

I would first like to thank my project supervisor, Professor Tor Onshus of the Department of Engineering Cybernetics at NTNU, for allowing me to work on such an open and interesting topic, and for providing valuable advice and guidance through the two last semesters. He has been quick to respond to problems with the project, and gave me with a sense of urgency when the project was lagging far behind schedule in march.

Among my student colleagues, Eirik Wold Solnør, Vegard Blomseth Johnsen and Henrik Rudi Haave have been particularly helpful during testing sessions. Over the last two semesters, Ole Magnus Siqveland and I have used the same robot platform for our projects. Through collaboration, we found a hardware setup that worked for both of us; a new shelf structure with compartments for the various hardware components.

I would like to thank the foreman, Terje Haugen, and apprentice Daniel Bogen at the mechanical workshop for building the new compartments and frames for the robot used in this thesis. Many thanks goes to the employees at the electronics workshop for allowing me to borrow tools and equipment, and providing some hints and tips.

I am very grateful to my parents and Andrea Myklebust for supporting me through the years at NTNU. Thank you!

Sincerely, Vegard Stjerna Lindrup

Contents

List of Acronyms **xiii**

List of Figures **xiii**

List of Tables **xv**

List of Acronyms **xvii**

List of Algorithms **xix**

1 Introduction **1**
1.1 About the Thesis 1
1.1.1 Long Term Goal and Previous Work 1
1.2 Implementation Overview 2
1.3 Thesis Structure 2

2 Task Outline and Planning **5**
2.1 Introduction 5
2.2 Task definition 5
2.3 Specification 5
2.4 Planning 5
2.4.1 Work Breakdown Structure 5

3 Background Theory **7**
3.1 Robotic Maintenance of Industrial Installations 7
3.1.1 Introduction 7
3.1.2 Potential Maintenance Tasks 7
3.1.3 Which Tasks Can Be Robotised? 7
3.1.4 Structural Maintenance and Environmental Considerations . 9
3.1.5 Production-specific Hazards 10
3.1.6 Implications for Robot Design 11
3.2 Robotic Maintenance Today 12
3.3 Modelling and Simulation 15

3.3.1 Some Terminology . . . 15
3.3.2 Robot Modelling . . . 15
3.3.3 Simulating in Gazebo . . . 15
3.4 ROS . . . 15
3.4.1 Introduction . . . 15
3.4.2 Important ROS Concepts . . . 16
3.4.3 An Overview of ROS-Related Tools . . . 17
3.4.4 Notable Robots Running ROS . . . 17
3.5 Software . . . 18
3.5.1 Qt . . . 18
3.5.2 PCL . . . 18
3.6 The Kinect Sensor . . . 18
3.7 Software Tools . . . 18
3.7.1 Point Cloud Library . . . 18
3.7.2 ROS . . . 18
3.7.3 Qt . . . 18
3.7.4 Current Research and Applications . . . 18
3.8 Introduction to Sensors in Autonomous Robots . . . 18
3.8.1 Depth Cameras . . . 18
3.8.2 Plannar Laser Sensors (LIDAR) . . . 20
3.8.3 Odometers . . . 20
3.8.4 Sensor Fusion . . . 20
3.9 Simultanious Localization and Mapping (SLAM) . . . 20
3.9.1 Introdunction to SLAM . . . 20
3.9.2 Hector SLAM . . . 20
3.9.3 RTAB-Map . . . 20
3.9.4 Octomap . . . 22
3.10 Navigation . . . 22

4 Implementation **25**
4.1 Introduction . . . 25
4.2 ROS Integration . . . 25
4.3 Modeling . . . 25
4.3.1 Introduction . . . 25
4.3.2 Physical Dimensions . . . 25
4.3.3 Coordinate Frames . . . 26
4.4 Simulations . . . 26
4.5 Motion Control . . . 26
4.6 ROS Nodes for Motion Control . . . 26
4.6.1 Velocity Command Sources . . . 26
4.6.2 Motor Control Card Firmware on XMEGA A3BU . . . 28
4.7 Operator Control Station (OCS) . . . 28

4.7.1 Graphical User Interface . . . 28
4.8 The Hand Held Remote Control - *Robot Leash* . . . 28
4.8.1 Connecting to the Robot . . . 28
4.9 Mapping . . . 30
4.10 Navigation . . . 30
4.10.1 Global Path Planning . . . 30
4.10.2 Local Path Planning . . . 30

5 Testing **31**
5.1 Testplan . . . 31
5.2 Results . . . 31
5.2.1 Simulations . . . 31
5.2.2 Live Testing . . . 31
5.3 Discussion . . . 31

6 Discussion **35**
6.1 Mapping . . . 35
6.2 Navigation . . . 35
6.3 Suitability for Offshore Maintenance . . . 35
6.3.1 Open Source Software and Security . . . 35

7 Conclusion **37**
7.1 Future Work . . . 37
7.1.1 Autonomous Non-Destructive Testing . . . 37
7.1.2 Large Scale Kinect Fusion - Kintinous . . . 37
7.1.3 Hardware . . . 38
7.2 Task Fulfilment . . . 39
7.3 Final Conclusion . . . 39

References **41**

Appendices

A Setting Up the Project **45**
A.1 Installation . . . 45
A.1.1 Equipment List . . . 45
A.1.2 Install Ubuntu . . . 45
A.1.3 Download ROS . . . 45
A.2 Configuring the Project . . . 45
A.2.1 Configuring the ROS Workspace . . . 45
A.2.2 Configuring the Bluetooth Connection . . . 45

B Troubleshooting **47**
B.1 Introduction . . . 47

B.2 ROS 47
B.2.1 ERROR: tf2:ExtrapolationException 47
B.3 Gazebo 48
B.3.1 Error [Node.cc.90] No namespace found 48
B.3.2 Dependency Issues When Installing *gazebo2* 48
B.4 Ubuntu 48
B.4.1 Ubuntu Freezes 48

List of Figures

1.1 System Concept. 3

3.1 Subsea 7's AIV. This is the first commercial autonomous inspection vehicle for subsea operations [pre] 12
3.2 Team HRP2-Tokyo's robot turning a valve during DARPA Robotics Challenge 2015 (Image credits: DARPA Robotics Challenge) 13
3.3 An early version of the maintenance robot "Sensabot", developed by National Robotic Engineering Center (NREC) (Image credits: NREC) . 14
3.4 A minimal Robot Operating System (ROS) graph. There are two nodes, none 1 and node 2. Node 1 publishes data, i.e. a topic, by the name `topic_1`. Node 2 can receive the data by subscribing to `topic_1`. 21
3.5 Awesome Image 21
3.6 Awesome Image 21
3.7 Conceptual illustration of a graph created by Real-Time Appearance-Based Mapping (RTAB-Map) over time $1 \leq t \leq 8$. A loop closure hypothesis was accepted at $t = 7$, as shown by the yellow arrow. Similarities between L_2 and L_7 is sufficient to accept this as a loop closure. 23

4.1 Robot model with frames for laser, Kinect, robot base and map. 27
4.2 Sensor input placed with correct transformations from *base_link*. . . . 27
4.3 Nodes and topics for motion control. 27
4.4 Velocity command ramping. The blue line represents commands entering *velocity_ramp*, while the red line shows the acceleration constrained output command. 29
4.5 A typical use case for "Robot Leash". 29
4.6 Detecting obstructions in 3d. 30

5.1 The "Asphalt" world in Gazebo. 32
5.2 An example of incorrect map merging. This case occurred in the "Asphalt" world simulated in Gazebo. 32
5.3 Nodes and topics for motion control. 33

5.4 Moving obstacle avoidance. The local cost map, shown as coloured spots on the occupancy grid, is based on real-time sensor data. 34

7.1 Worn omniwheel 38

List of Tables

3.1 Some tasks with potential to be ”robotized”, and the corresponding robot category[PBB11]. 8

List of Acronyms

AI Artificial Intelligence

AIV Autonomous Inspection Vehicle

CLM Concurrent Localization and Mapping

CP Cathodic Protection

DRC DARPA Robotics Challenge

Fraunhofer IPA Fraunhofer Institute for Manufacturing Engineering and Automation

GUI Graphical User Interface

HMI Human-Machine Interaction

HSE Health, Safety and Environment

IFR International Federation of Robotics

IO Integrated Operations

ISS International Space Station

ITK Department of Engineering Cybernetics

LIDAR LIght Detection And Ranging

MIMROex Mobile Inspection and Monitoring Robot, experimental

NCS Norwegian Continental Shelf

NDT Non-destructive Testing

MIT Massachusetts Institute of Technology

NUI Natural user interface

OCS Operator Control Station

O&G Oil & Gas

PCL Point Cloud Library

PR Personal Robot

PWM Pulse Width Modulation

RBI Risk Based Inspection

ROS Robot Operating System

ROV Remotely Operated Vehicle

RPAS Remotely Piloted Aerial System

RTAB-Map Real-Time Appearance-Based Mapping

SDF Simulation Description Format

SIFT Scale-invariant feature transform

SIM Structural Integrity Management

SLAM Simultaneous Localization And Mapping

SSD Solid State Drive

SURF Speeded Up Robust Features

STAIR Stanford AI Robot

UAV Unmanned Aerial Vehicle

URDF Unified Robot Description Format

VR Virtual Reality

List of Algorithms

Chapter 1
Introduction

Introduction

1.1 About the Thesis

1.1.1 Long Term Goal and Previous Work

The Topic - Mobile Autonomous Robot

The robot system that was used in this project, has been developed over the course of many preceding master and specialization projects. The long term goal of these projects is to develop a mobile autonomous robot for maintenance and inspection on topside offshore installations. The topic is given by Professor Tor Onshus at Department of Engineering Cybernetics (ITK). A description of this topic[1], suggests some possible applications for such a robot:

- The robot could serve in a supporting role as a part of Integrated Operations (IO).
- It can also be used to prepare an unmanned topside offshore installation before the arrival of a maintenance crew, by performing safety checks and preparing the helicopter landing pad.
- Allow personnel to perform remote inspection and maintenance through telepresence.
- In combination with Virtual Reality (VR), the robot could be used for training purposes.

[1] http://folk.ntnu.no/onshus/Oppgaver.htm

Building the Mobile Platform

Simultanious Localization and Mapping

1.2 Implementation Overview

1.3 Thesis Structure

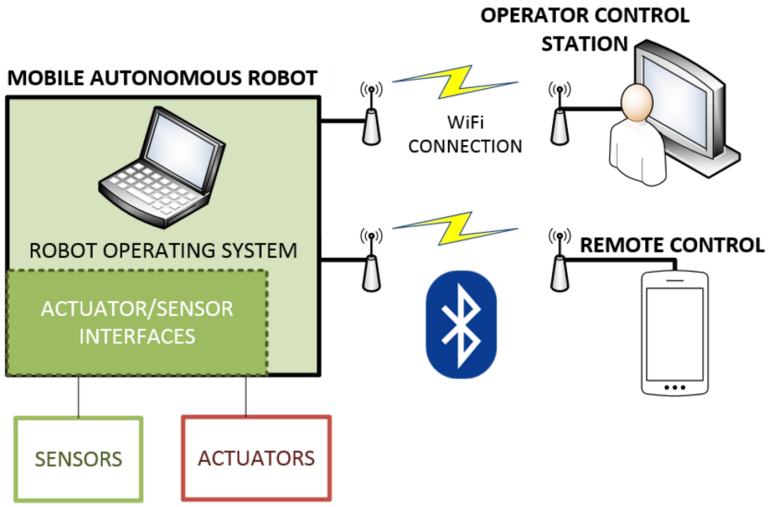


Figure 1.1: System Concept.

Chapter 2
Task Outline and Planning

2.1 Introduction

The original problem description given at the beginning of this semester is very open. A significant part of the project is oriented towards

2.2 Task definition

2.3 Specification

2.4 Planning

2.4.1 Work Breakdown Structure

Chapter 3
Background Theory

3.1 Robotic Maintenance of Industrial Installations

3.1.1 Introduction

This project is a small step towards a larger long-term goal concerning robotic maintenance on topside offshore installations. This section puts the following background theory, and the implementation described in chapter 4, into the context of the long-term goal.

This section will introduce how topside maintenance is performed today, how these maintenance tasks could be robotised. The section is concluded with a discussion on how well modern robotics is suited for the task.

3.1.2 Potential Maintenance Tasks

Hidden failure modes: PFD: What is the probability that a device (Fire detector, shut down valve, etc.) will fail when needed? Solution: Periodic maintenance.

3.1.3 Which Tasks Can Be Robotised?

Over the last decade, the Oil & Gas (O&G) industry has shown an increased interest in the potential benefits of automating the normal operation of remote offshore installations. It is now clear that robotics has several potential applications in process plants, and particularly in remote O&G installations. Current research on plant automation is mainly motivated by two factors[KMP15] [AS12]:

- **HSE** - Reduced risk exposure for personnel and environment.
- **Efficiency** - Accomplish more with less effort, resources and time. This means cost reduction by keeping downtime to a minimum with the least amount of effort.

Robot Applications Including Categorization		
Category	Robot	Robot Task/Activity Description
B	Pipeline rigging robot	To autonomously load and offload pigs into pipelines.
C	Boat handling robot < 500 kg	To transfer personnel and loads below 500 kg to and from boats.
D	Boat handling robot > 500 kg	To transfer loads above 500 kg to and from boats.
B	Mobile universal service robot version 1	To perform "buddy" roles; carrying, holding, lifting, personal safety monitoring etc.
C	Mobile universal service robot version 2	To autonomously perform task not involving manipulation of the process or facilities.
D	Mobile universal service robot version 3	To autonomously perform tasks involving manipulation of the process or facilities.
D	Treatment/Inspection robot	To autonomously perform inspection/treatment(painting) tasks of structures/vessels or facilities.
A	Domestic service robot	To autonomously perform floor cleaning, catering, laundry handling, storage handling and logistics activities.

Table 3.1: Some tasks with potential to be "robotized", and the corresponding robot category[PBB11].

An additional overarching driving factor is that O&G fields are becoming more difficult to reach. As O&G fields in shallow waters are depleted, production is moved to deeper waters. This complicates the extraction process and reduces the profit margin. A solution to this challenge is to increase efficiency through further automation.

A feasibility study performed by researchers from Fraunhofer Institute for Manufacturing Engineering and Automation (Fraunhofer IPA) identified several topside production tasks and ranked them according to their resource demand. Based on the identified tasks, the study went on to describe a set of specific tasks, and then assess how easy or hard it would be to "robotize" these tasks. Table 3.1 associates a set of tasks with different robot categories, as well as how easy or hard the process of "robotizing" the activity is expected to be. The difficulty levels are described

with the letters ”A” through ”D”, where ”A” is described as of the shelf robotics, which makes ”robotizing” easy. Activities associated with the letter ”D” on the other hand, cannot be be ”robotized” with either current or near future technology even if doing so would be beneficial [PBB11]. Note that the paper in question is from 2011, and the difficulty of these tasks may have changed. This is particularly true in the domain of visual sensors, given the bloom of 3d sensor technology over the last six years. Some further elaboration on inspection activities and environmental considerations is given in the next subsection.

3.1.4 Structural Maintenance and Environmental Considerations

Corrosion

Offshore installations are regularly, if not continuously, exposed to harsh weather conditions in the form of wind and seawater. Presence of seawater, either through direct contact or in the form of drops and vapor, forms a very corrosive environment. The offshore and marine environment is classified as the most corrosive environment in ISO 12944[ER12a]. It is essential to provide countermeasures to ensure safe and reliable operation over the lifetime of the installation. Common corrosion prevention methods are[ER12a]:

- Sacrificial Anodes.
- Cathodic Protection (CP) in the form of a DC-current.
- Protective coating.

In terms of maintenance, the sacrificial anodes can be subjected to periodic inspections and replacements, which could be done by a robot. CP can more easily be implemented with automated self tests, and should normally not require any inspections and maintenance[ER12a]. Application of protective coating should ideally be applied in the controlled environment of a workshop. If protective coating is to be applied at sea, one should strive to make the conditions as favourable as possible.

Structural Fatigue

Waves, wind, water currents and other forces subject offshore installations to structural stress. To keep the offshore installations from failing in these conditions, they may be subjected to a Risk Based Inspection (RBI) regime. In brief, RBI is a strategy where inspection and maintenance programs are developed based on which risk factors an installation is exposed to. In an automated maintenance program, an autonomous robot could perform inspections of the structure and generate reports based on risk factors such as[ER12b]:

Marine growth at sea level Marine growth will increase the diameter of supporting legs at sea level, thus increasing structural loads caused by waves, wind and water currents.

Corrosion Assess the seriousness of a corrosion attack through Non-destructive Testing (NDT).

Scour Scour around the platform legs could reduce a platforms ability to withstand structural loads. This is only applicable to non-floating installations.

A maintenance planner can then plan a maintenance campaign based on data from the robot in combination with knowledge on the platforms design, age and exposure to the environment.

3.1.5 Production-specific Hazards

O&G production has several inherent hazards, and an unwanted incident may have serious implications for Health, Safety and Environment (HSE) and production uptime. The Piper Alpha incident serves as a worst case example of the consequences of an explosive ignition of a hydrocarbon leak followed by an escalating fire. This section will briefly discuss some of the most significant hazards on an offshore oil and gas production plant.

Hydrocarbon leaks

Hydrocarbon leaks do occur on a regular basis. Over a four year period from 2006 to 2010, seven leaks larger than $1kg/s$ of either oil or gas/two-phase occurred in the Norwegian sector. No such leaks have ignited on the Norwegian Continental Shelf (NCS) since 1992. Of all the leaks which occurred in the same area, NCS, the majority was caused by human intervention[Vin14]. This could imply that a reliable robotic system may reduce the number of leaks.

Fire

Critical fire loads[1] on offshore facilities are usually caused by uncontrolled flow of hydrocarbons. The most serious of such releases is a blow-out. Risk reducing measures focus on four areas[Vin14]:

Leak prevention - Use equipment and assembly methods which minimize risk.

[1]Fire load can be defined as the amount of combustible material in a given area (Ref. https://en.wiktionary.org/wiki/fire_load).

Leak detection - Fire & gas detection, emergency shut-down systems and blow-down systems.

Ignition prevention - Inspection and maintenance and Ex-approved equipment.

Escalation protection - Installation layout and sectioning. Fire and gas protection systems.

Explosions

Explosion protection is usually built into the equipment and structure. In [Vin14], there are no obvious ways a robot could provide additional explosion beyond e.g. leak detection, inspection and maintenance.

3.1.6 Implications for Robot Design

A mobile robot operating on a normally unmanned platform in a harsh environment, implies that it is subjected to many of the same design philosophies that apply to subsea equipment.

Because of the risk of explosive atmospheres in an offshore production environment, an offshore robot operating under EU or EEA legislation will also be subject to the ATEX (ATmosphères EXplosibles) directive. Such a robot will most likely carry ignition sources such as batteries packed with energy and perhaps even welding equipment. A central ATEX requirement is to perform a risk assessment. As outlined by ATEX 2014-34-EU Guidelines[ATE], such a risk assessment is usually performed in four steps:

1. **Hazard identification** What can go wrong? Identify possible ignition sources, and the probability of explosive atmospheres.

2. **Risk estimation** Estimate the probability of an unwanted occurrence (e.g. an explosion), and the associated consequences.

3. **Risk evaluation** Evaluate the identified risk in context of acceptable risk, and decide if the design should be altered or if additional barriers should be installed. Barriers could either mitigate the consequences of an explosion, or reduce the possibility of ever having an explosion.

4. **Risk reduction option analysis** Identify possible risk reduction measures, e.g. barriers and design changes. A cost-benefit analysis can be performed in accordance with the ALARP-principle.

The on-board embedded computer hardware and software should be designed for robustness, fault tolerance and endurance. If a failure occurs, corrective actions will most likely be both difficult and expensive.

Resistance to corrosion and toxic environments should also be taken into account.

3.2 Robotic Maintenance Today

Gas leak detection: [SWB+14]. DARPA robotic challenge. Industrial ROS.

Trends and Potential

The typical pre-programmed assembly robots still dominate the robotic market. They are usually found in manufacturing plants and large scale production facilities[ifr], e.g. the automotive industry, where they perform dull, tedious tasks much faster and with higher accuracy than people. A notable trend in modern robotics is increased human-robot collaboration[Bog16]. Many new robots are being build for the human workspace, both in terms of safety and collaborative functionality. This trend is a step along the way of moving robots out of the controlled environment of a factory floor, and into the real world where a high degree of autonomy is required.

A report by Metra Martech[GC11], a market research firm referenced to by International Federation of Robotics (IFR)[2], points to three areas with a high potential for robotic applications:

- Dangerous jobs, e.g. handling dangerous materials or work in high risk environments.
- Jobs that are economically infeasible in a high wage economy.
- Work which is impossible or highly inconvenient for humans, e.g. space exploration, subsea maintenance or assembly of heavy components.

All of these factors motivate the development of robots for autonomous robotic maintenance.

Subsea Maintenance and Inspection

Subsea maintenance is perhaps the field that have seen the greatest advancements in autonomous inspection and

[2]http://www.ifr.org/robots-create-jobs/

maintenance. As offshore installations are moved to the seabed, maintenance and inspection has become a significant challenge. This has resulted in a widespread use of Remotely Operated Vehicles (ROVs). Recent developments in other fields, e.g. computer vision, human-robot collaboration and machine learning, has resulted in new Autonomous Inspection Vehicles (AIVs) and Autonomous Underwater Vehicles (AUVs) capable of performing inspection and simple maintenance tasks autonomously[JWA+12][RCR+15]. A driving factor behind the transition from ROVs to AUVs is cost reduction through increased offshore campaign efficiency.

Disaster Responce

Robots in disaster response, relief and recovery solve many of the same problems faced by maintenance robots. Disasters, such as the tsunami which struck Japan in 2011, proved that much work needs to be done, both in terms of technical capabilities and logistical issues related to deployment and response times. The tsunami resulted in three core meltdowns at the Fukushima Daiichi Nuclear Power plant.

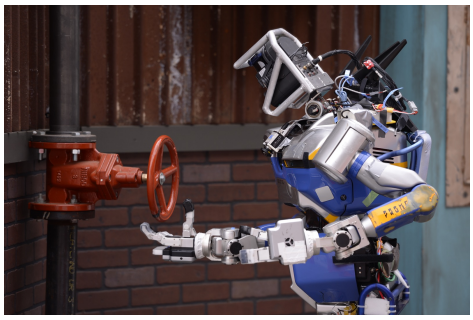


Figure 3.2: Team HRP2-Tokyo's robot turning a valve during DARPA Robotics Challenge 2015 (Image credits: DARPA Robotics Challenge)

Many of the robots which were deployed at the Fukushima Power Plant were already ageing, and the operators had to receive training before deployment, thus increasing the response time[KFO12]. A paper from Japan Atomic Energy Agency[KFO12] highlights how the lack of stakeholder involvement could have been the cause of long response times. The same paper points out that the robots were developed for the sake of development, and not with emergency response as the main purpose[KFO12].

DARPA Robotics Challenge (DRC)[DRC] was launched in response to the Fukushima

disaster of 2011. The purpose of the competition is to accelerate innovation, research and development in robotics for disaster response in cases where humans cannot operate. Some of the tasks the competitors faces in 2015 include valve turning, traversing rubble and driving a vehicle through a course before egressing out of the vehicle.

Topside Offshore and Onshore Robotic Maintenance

Today, autonomous and teleoperated inspection and maintenance is usually only found at subsea installations. Topside installations on the other hand are still maintained and inspected manually, with some notable exceptions. Small Unmanned Aerial Vehicles (UAVs) or Remotely Piloted Aerial Systems (RPAS) have become commonplace over the last decade. On topside installations, they are being used for visual inspection of inaccessible structural parts such as flare stacks or the exterior of oil rigs.

Figure 3.3: An early version of the maintenance robot "Sensabot", developed by National Robotic Engineering Center (NREC) (Image credits: NREC)

Among the research groups working on robotic maintenance for O&G, ABB, Fraunhofer IPA and NREC at Carnegie Mellon University stand out as major contributors to the field.

NRECs contribution, Sensabot, is a remotely operated inspection robot designed for harsh and remote environments[dep12]. It is not designed to be autonomous, but rather as a tool to move personnel from hazardous environments to safe remote control rooms. Sensabot mark II will be certified for zone 1 explosive environments. This year (2016), the plan is to test the robot on site at the Kashagan field in Kazakhstan[PSM+16].

Fraunhofer IPA[3] has developed a robot, called Mobile Inspection and Monitoring Robot, experimental (MIMROex), with capabilities which are quite similar to the prototype used during the work on this thesis. MIMROex is equipped with a camera for visual inspections as well as microphones, vibration and sensors for fire and gas

[3] http://www.ipa.fraunhofer.de/en.html

detection. It is also certifiable in accordance with the explosion protection standard IEC 60079[MIM].

Another effort towards robotic maintenance is the ARGOS challenge (Autonomous Robot for Gas and Oil Sites). The purpose of the challenge is to promote innovation, understanding and awareness towards robotic maintenance of O&G sites in harsh environments[ARG].

3.3 Modelling and Simulation

3.3.1 Some Terminology

Coordinate Systems and Poses

Robot Joints

All links are connected to each other by joints.

Coordinate systems are essential in the field of robotics.

3.3.2 Robot Modelling

3.3.3 Simulating in Gazebo

3.4 ROS

3.4.1 Introduction

The ROS is a collection of software libraries, tools and drivers intended for robot software development. A ROS installation can be tailored to meet the demands of a wide range of robots with varying complexity. ROS is usually installed in the form of an already built Debian-package. These packages are only compatible with a few versions of Ubuntu which are specified on the ROS homepage. When installed and configured, ROS will run on top of Linux, and can be perceived as and extention of Linux itself. Installing ROS from source is possible, but not recommended [ROSc].

Roots of ROS can be traced back to Stanford University at the beginning of the 2000s. At Stanford, several robotics software frameworks, including Stanford AI Robot (STAIR) and the Personal Robot (PR) program, were created to provide dynamic, flexible and well tested foundations for further robot development and research. In 2007, a nearby start-up company and robot incubator, Willow Garage, sought to build upon these concepts, and initiated a collaborative and open development process of a new software framework. This framework eventually became ROS[ROSb][QGS15]. The framework can be used under the BSD open-source license[**?**]. Today, ROS comes in many forms and comprise hundreds of advanced packages, algorithms and

drivers, making it applicable for hobbyists, industrial automation, research and everything in between.

3.4.2 Important ROS Concepts

The following descriptions are included in order to provide a complete, self-contained description of the project implementation. Similar descriptions can be found on the official ROS website[4], as well as in any book on ROS (for example [QGS15]).

The ROS Graph

A ROS system comprise a set of small programs that communicate with each other through messages. These programs become nodes in the ROS graph. The nodes communicate with each other by publishing and subscribing to topics that form the edges of the graph. A topic must have the format of one of the specific data types provided by ROS. For example, a node which receives temperature data from a thermometer, may publish the data as a topic on the ROS system with the type `sensor_msgs/Temperature`. There are many other data formats, e.g. velocity messages, `geometry_msgs/Twist`; images, `sensor_msgs/Image`; odometry messages, `nav_msgs/Odometry` and so on. Each node in the graph are typically POSIX processes, and the edges are TCP connections[QGS15]. A minimal example of a graph is shown in figure 3.4.

roscore

`roscore` is an essensial part of any ROS system as it enables nodes to communicate with each other. An instance of `roscore` must be started before launching any nodes. When a node is started, it will inform `roscore` of which topics it publishes and which topics it wish to subscribe to. Then, `roscore` will provide the information which allows the node to form a peer-to-peer connection to other nodes.

Project Structure and the *catkin* Build System

The source code in a ROS system is organized into packages. Each package provides a specific functionality to the system. Some packages can be downloaded and installed from a remote repository, while other packages will be created by the in-house developers for their specific robotic system. In this project, locally created ROS-packages were placed into a *catkin workspace*. This workspace contains the original source code and build specifications. Details are provided in chapter 4. As presented in[ROSa], a general workspace structure is as follows:

```
workspace_folder/        -- CATKIN WORKSPACE
```

[4] http://www.ros.org/

```
src/                    -- SOURCE SPACE
  CMakeLists.txt        -- 'Toplevel' CMake file, provided by catkin
  package_1/
    CMakeLists.txt      -- CMakeLists.txt file for package_1
    package.xml         -- Package manifest for package_1
  ...
  package_n/
    CMakeLists.txt      -- CMakeLists.txt file for package_n
    package.xml         -- Package manifest for package_n
```

A ROS project will usually utilize the catkin build system.

3.4.3 An Overview of ROS-Related Tools

Robot Modelling In URDF

Unified Robot Description Format (URDF) is an XML-like format for describing robots. The robot description is made up of links and joints. Each link description contains information of its shape, inertial tensor, collision boundaries. The links are connected to each other by joints.

Visialization in `rviz`

`rviz` is an invaluable tool for visualizing on-line robot behaviour. Simply put, `rviz` is created to visualize what the robot sees, and how it plans ahead. Many of the images in the following chapters are from `rviz`.

Simulation in Gazebo

3.4.4 Notable Robots Running ROS

PR2 - Personal Robot 2 PR2 is one of the first robots designed to run ROS [QGS15], and also one of the most advanced and capable robots with ROS today.

TurtleBot TurtleBot is a cheaper ROS-ready alternative to PR2.

Robonaut 2 Robonaut 2, a dexterous humanoid robot, currently resides within the International Space Station (ISS) 400 km above the earth's surface. In 2014, a SpaceX Dracon capsule brought ROS as well as a pair of legs for Robonaut up to the ISS[ROSd]. Robonaut is designed for research on human-robot collaboration in space, and human-like tasks. For more information, follow **this link**[5] to a talk on ROS in space from ROSCon 2014.

[5] https://vimeo.com/106993914

Being the first robot with ROS to be launched into space,

Example of an industrial robot with ROS goes here! TODO!!!!!!!!!!

3.5 Software

3.5.1 Qt

3.5.2 PCL

3.6 The Kinect Sensor

3.7 Software Tools

3.7.1 Point Cloud Library

3.7.2 ROS

3.7.3 Qt

3.7.4 Current Research and Applications

3.8 Introduction to Sensors in Autonomous Robots

3.8.1 Depth Cameras

Different Methods for Depth Perception

A depth camera can be described as a regular color video camera with the ability to create spatial images. In the context of this thesis, a depth camera can more precisely be described as a RGB-D camera, where the letters RGB-D are short for red, green, blue and depth. In a regular RGB camera, a spatial scene will be projected onto a rectangular pixel grid where each pixel contains intensity values for red, green and blue colors. These pixel values represents the detected scene. A major problem with RGB cameras is the significant loss of information. The information loss is mostly a consequence of 3d to 2d projection and digital quantization. RGB-D cameras have the means to reduce this information loss by mapping the pixel values to spatial coordinates. The atomic parts in 3d images are usually represented as point clouds or cubic volumes, also known as voxels.

Different variations of depth cameras will usually fall into one of two categories: active or passive. Passive sensors perceive the surroundings as it is, without actively interfering with the environment as a part of the sensing process. A typical passive RGB-D sensor is the stereo camera. Stereo cameras use a stream of synchronized image pairs to perceive depth. The image pairs are displaced along the horizontal

axis, and the depth information is extracted by searching for mutual information in the image pairs. How far the information is displaced from the left to the right image is directly related to how far away from the camera the information source is located.

Active sensors depend on some form of projection onto the surroundings. For depth cameras, the projection is usually in the form of laser or infra red light. In RGB-D cameras it is essential that the projected light is distinguishable from the visible spectrum. The Kinect sensor used in this project is an example of an active RGB-D sensor. A proper introduction to the Kinect, will follow shortly.

Natural User Interfaces - Origin of the Kinect

The idea behind a Natural user interface (NUI) is to make Human-Machine Interaction (HMI) as seamless and natural as possible. A NUI allows the user to communicate without tools such as a keyboard or a mouse. For decades, NUIs have only existed as ideas, science fiction or research projects. This has changed dramatically over the last ten years, and NUIs can now be considered to be ubiquitous. Today, the most common form of NUIs is the touch screen found in smart phones and tablets.

The Microsoft Kinect sensor was initially designed as a NUI for the Xbox 360 gaming console. The sensor allows users to use gestures and sounds to play console games. Later on, Microsoft has released SDKs, enabling developers to create NUI applications for for Windows.

Kinect for Xbox 360

Kinect for Xbox 360 is the RGB-D sensor used in this project. The device was initially intended as a NUI for gaming and office applications, and was the first consumer grade sensor to utilize structured light. Possible use cases were inspired by early NUI research at Massachusetts Institute of Technology (MIT) and, later on, the science fiction movie Minority Report, where Tom Cruice interacts with a computer by using hand gestures [WA12]. The Kinect sensor is equipped with a depth sensor, a regular color camera, a microphone array and a tilt motor. The color camera in combination with the depth sensor forms what is usually referred to as a RGB-D sensor, i.e. a combined color and depth camera (figure 3.6). This feature, combined with the relaticely low cost and accessability of the sensor has contributed to make the Kinect very popular in research projects related to Simultaneous Localization And Mapping (SLAM) and robotics. In the three first years since it's release in 2010, over 3000 papers in well-known journals and proceedings were devoted to research on the Kinect sensor. Roughly 500 of these papers focused on SLAM or 3d reconstruction[BMNK13]. Some of the other papers focused on some of the weaknesses with the sensor, such as detection of glass surfaces and having several sensors in the same area.

Today, the the Kinect for Xbox 360 has been succeded by the Kinect for Xbox One, and is now considered to be a legacy device. Those considering to use the legacy Kinect should be aware of that it is becoming increasingly difficult, if not already impossible, to get hold of a new Kinect for Xbox 360.

3.8.2 Plannar Laser Sensors (LIDAR)

A plannar laser sensor, known as e.g. laser proximity sensors or laser radars, can all be referred to as LIDARs.

Scanning Laser Range Finder, URG-04LX-UG01

3.8.3 Odometers

3.8.4 Sensor Fusion

3.9 Simultanious Localization and Mapping (SLAM)

3.9.1 Introdunction to SLAM

SLAM, also known as Concurrent Localization and Mapping (CLM), is a class of solutions to the problem of determining an agents location and pose in an unknown environment, while simultaneously mapping the same environment.

3.9.2 Hector SLAM

3.9.3 RTAB-Map

RTAB-Map is developed by IntRoLab at Université de Sherbrooke in Canada. It is a SLAM system developed for long term operations in large environments. The system is also intended to handle the ”kidnapped robot-problem”, i.e. multi-session mapping. This is useful whenever the robot is shut down and moved to an unmapped part of the same area, where it will start a new mapping session. RTAB-Map is the core feature that has been integrated into the robot described in this thesis. Some factors which motivated the use of RTAB-Map are:

- It is a SLAM method which requires an RGB-D sensor, for example a Kinect. The problem description for this project requires a vision based solution.
- RTAB-Map has a ROS wrapper, `rtabmap_ros`, which eases the process of integrating it with the mobile robot.
- It includes 3d obstacle detection.

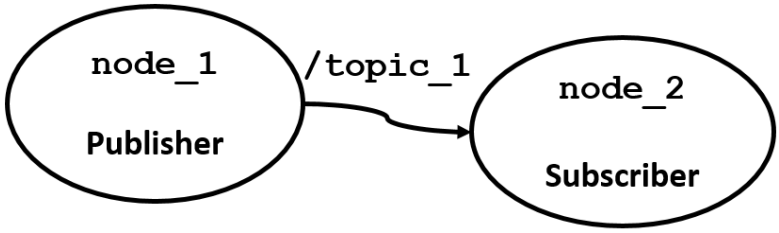


Figure 3.4: A minimal ROS graph. There are two nodes, none 1 and node 2. Node 1 publishes data, i.e. a topic, by the name `topic_1`. Node 2 can receive the data by subscribing to `topic_1`.

Figure 3.5: Awesome Image

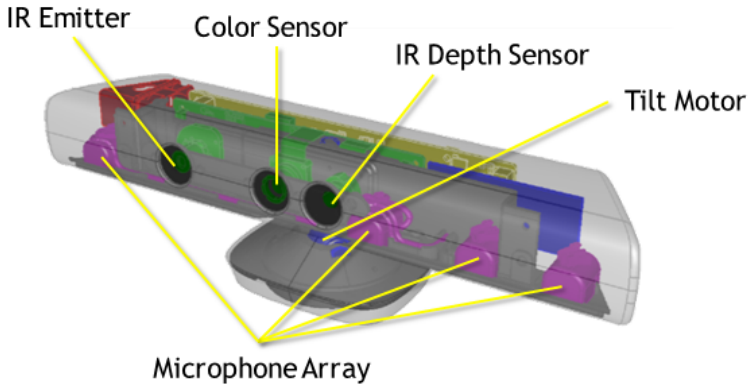


Figure 3.6: Awesome Image

- It has a memory management system intended for large scale multi-session mapping.
- RTAB-Map can be used for object detection. This can be done by linking RTAB-Map to OpenCV and the non-free feature detectors Scale-invariant feature transform (SIFT) and Speeded Up Robust Features (SURF).

The source code and ROS wrapper is currently maintained, and new features and bug-fixes are added regularly. RTAB-Map has two distinctive solutions to the SLAM problem: Visual loop closure detection and a memory management system for large data sets. The following paragraphs provides an overview of how RTAB-Map works. Detailed descriptions of the loop closure detection and memory management approach is provided in [LM13], while the SLAM method is presented in [LM14]. Further details can be found on the project's Github page[6].

Graph Based Mapping

RTAB-Map uses a graph structure with nodes and edges to represent the map. New nodes are continuously added to the systems working memory as time passes. In this method, the graph edges are referred to as *links*. There are two types of links: neighbour links and loop closure links. Each node is a location in the map, and the links contain geometrical transformations between the node locations. Figure 3.7 illustrated the graph concept.

On-line Mapping of Large Environments

3.9.4 Octomap

3.10 Navigation

[6] http://introlab.github.io/rtabmap/

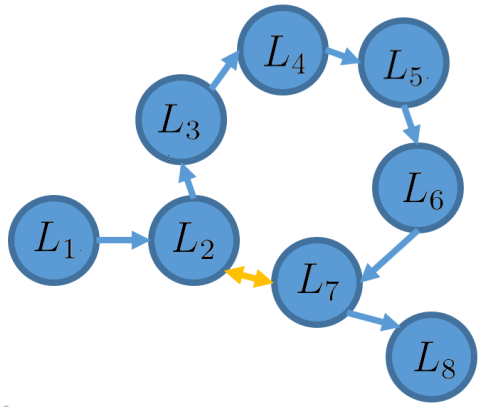


Figure 3.7: Conceptual illustration of a graph created by RTAB-Map over time $1 \leq t \leq 8$. A loop closure hypothesis was accepted at $t = 7$, as shown by the yellow arrow. Similarities between L_2 and L_7 is sufficient to accept this as a loop closure.

Chapter 4
Implementation

4.1 Introduction

4.2 ROS Integration

Implementation procedure for mobile robot:

- Decide on ROS message interface.
- Write interfaces for the motor drivers.
- Create a description of the physical structure and properties of the robot in URDF.
- Extend the model to enable simulation in Gazebo.
- Publish coordinate transform data via *tf* and visualize it in rviz.
- Add sensors, with driver and simulation support.
- Apply algorithms for navigation and other functionality.

4.3 Modeling

4.3.1 Introduction

4.3.2 Physical Dimensions

The inertia tensor:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{4.1}$$

Inertia tensor for a solid, uniform cylinder where the radius r is measured in parallel to the $x - y$ plane, and h is parallel to the z axis:

$$I_{cylinder} = \frac{1}{12} m \begin{bmatrix} (3r^2 + h^2) & 0 & 0 \\ 0 & (3r^2 + h^2) & 0 \\ 0 & 0 & r^2 \end{bmatrix} \tag{4.2}$$

Inertia tensor for a solid, uniform cuboid. The subscript of l indicates which axis l is measured along:

$$I_{cuboid} = \frac{1}{12} m \begin{bmatrix} (l_y^2 + l_z^2) & 0 & 0 \\ 0 & (l_x^2 + l_z^2) & 0 \\ 0 & 0 & (l_x^2 + l_y^2) \end{bmatrix} \tag{4.3}$$

4.3.3 Coordinate Frames

4.4 Simulations

Robot simulation was done in Gazebo, a simulation tool with good interfaces to ROS. The same ROS graph was used for both the simulated and real version of the robot, except from the sensors and actuators, and some minor parameter changes.

4.5 Motion Control

4.6 ROS Nodes for Motion Control

4.6.1 Velocity Command Sources

There are four ways to control the robot:

- Local keyboard input.
- Wireless teleoperation from the Operator Control Station (OCS).
- Wireless teleoperation from a handheld Bluetooth device.
- Commands from the navigation stack in ROS.

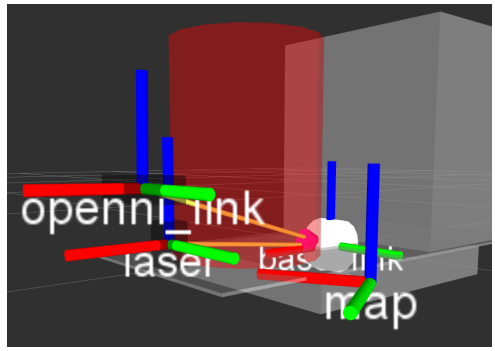


Figure 4.1: Robot model with frames for laser, Kinect, robot base and map.

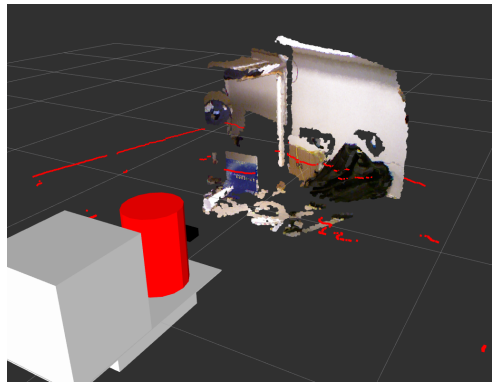


Figure 4.2: Sensor input placed with correct transformations from *base_link*.

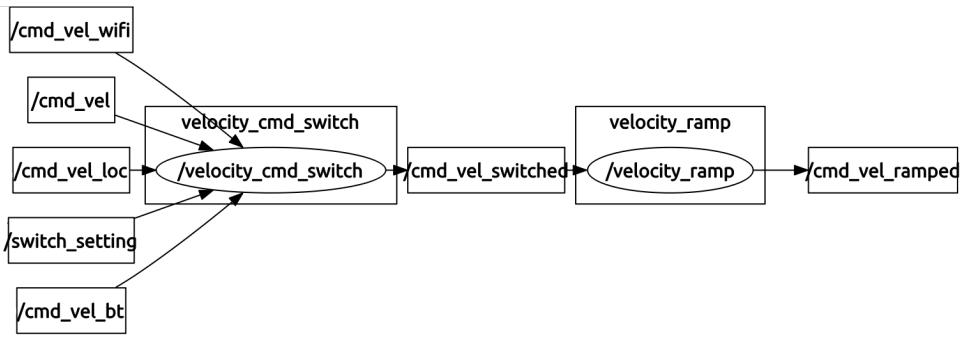


Figure 4.3: Nodes and topics for motion control.

4.6.2 Motor Control Card Firmware on XMEGA A3BU

XMEGA A3BU is an evaluation board developed by Atmel. The board has been used in earlier projects on this robot. The implementation presented here, is based on Petter Aspunviks implementation [Asp13]. The firmware will now receive velocity commands based on the `geometry_msgs/Twist` message format in ROS, and translate these into the command format used by each motor. Speed settings for each motor is based on Pulse Width Modulation (PWM).

There were two requirements for this implementation:

1. When velocity commands from the operating system are either absent or incomplete, the robot shall stop.
2. The program shall translate linear and angular velocity commands into wheel commands.

The connections are the same as in [Asp13], except for the installation of more secure connections under the robot. The old connections were insecure, and the risk of short circuits was substantial.

4.7 Operator Control Station (OCS)

The OCS allows an operator to control and monitor the robot through a graphical user interface. `MainWindow`

4.7.1 Graphical User Interface

A Qt-based Graphical User Interface (GUI)...

4.8 The Hand Held Remote Control - *Robot Leash*

Because the OCS is only partially implemented, an operator will not have access to all the features on the robot. In addition, as a safety precaution a person should be close to the robot at all times, and be ready to pull the plug. Furthermore, it is hard to control a moving robot through the on-board keyboard. These problems were countered by the Android-based remote control, *Robot Leash*.

4.8.1 Connecting to the Robot

1. The first screen after scanning for devices. There is no device filtering, and the user can select any device, but only connect through a specific service.

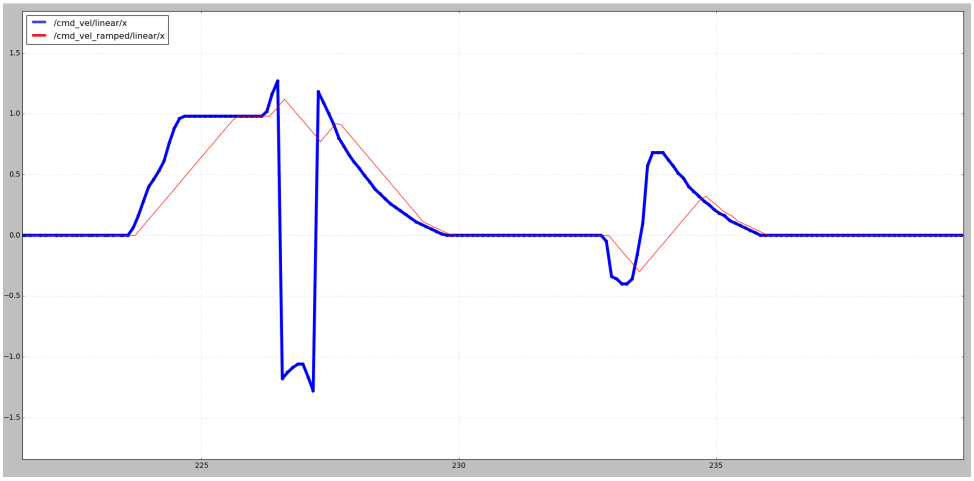
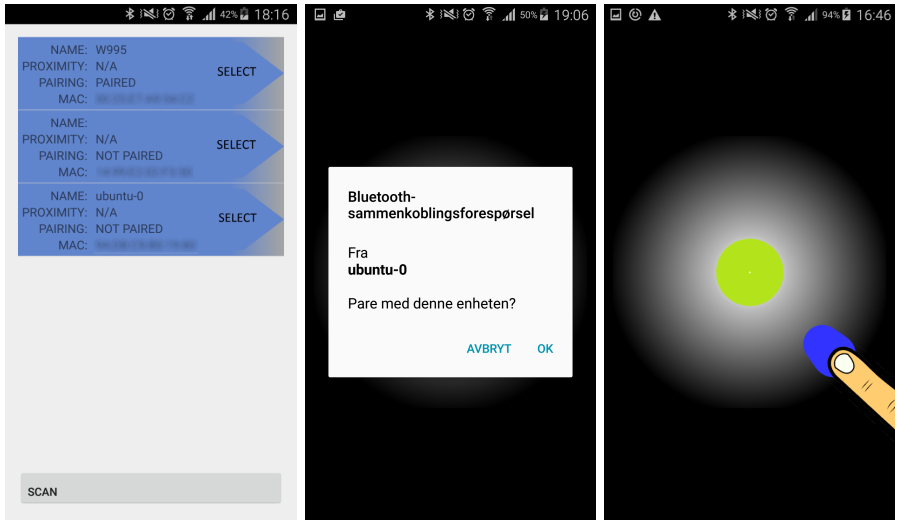


Figure 4.4: Velocity command ramping. The blue line represents commands entering *velocity_ramp*, while the red line shows the acceleration constrained output command.

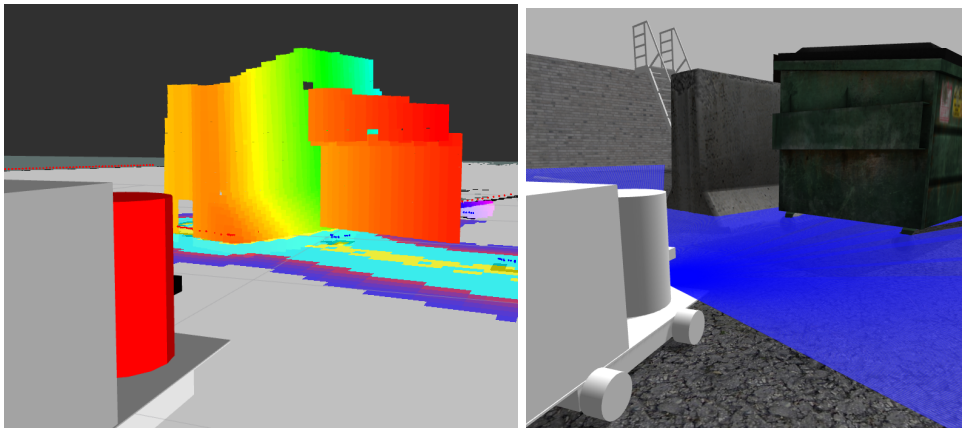


(a) First activity with device list.

(b) The user is prompted to pair with the robot.

(c) Controlling the robot with the stick.

Figure 4.5: A typical use case for "Robot Leash".



(a) A point cloud representation of the obstruction. Notice how the local costmap is based on the detected point cloud.

(b) The obstruction in the Gazebo simulator. Notice how the LIDAR only detects the wheels below the container.

Figure 4.6: Detecting obstructions in 3d.

2. After selecting a device which provides the correct service, the user will be prompted to pair the devices.

3. The smartphone and the robot is now paired, and velocity commands from the blue control stick are passed to the robot via Bluetooth.

http://developer.samsung.com/technical-doc/view.do?v=T000000117

4.9 Mapping

4.10 Navigation

4.10.1 Global Path Planning

4.10.2 Local Path Planning

Obstruction Detection

Chapter 5
Testing

5.1 Testplan

5.2 Results

5.2.1 Simulations

5.2.2 Live Testing

Safety Features

Loop Closure Detection

Multi Session Mapping

Navigating an Obstacle Course

Avoiding Moving Obstacles

5.3 Discussion

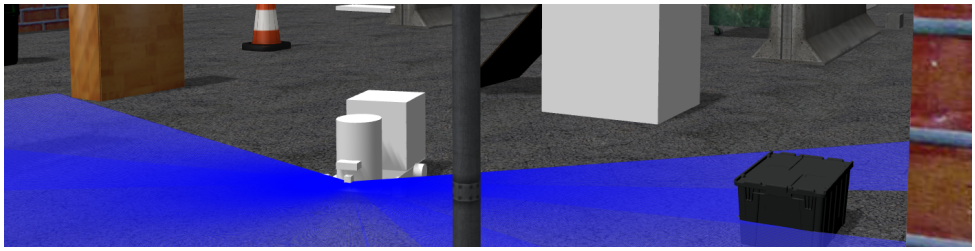


Figure 5.1: The ”Asphalt” world in Gazebo.

Figure 5.2: An example of incorrect map merging. This case occurred in the ”Asphalt” world simulated in Gazebo.

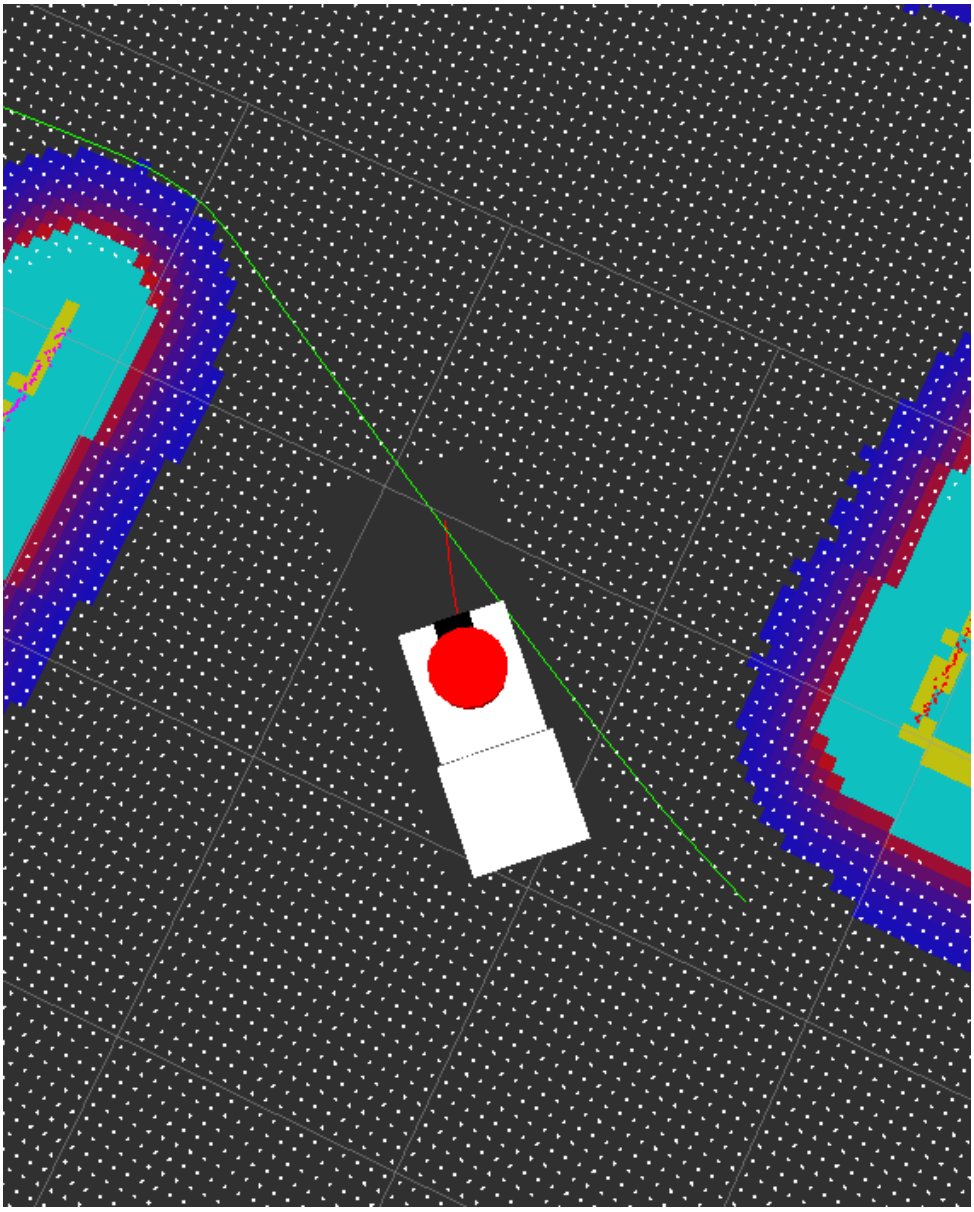
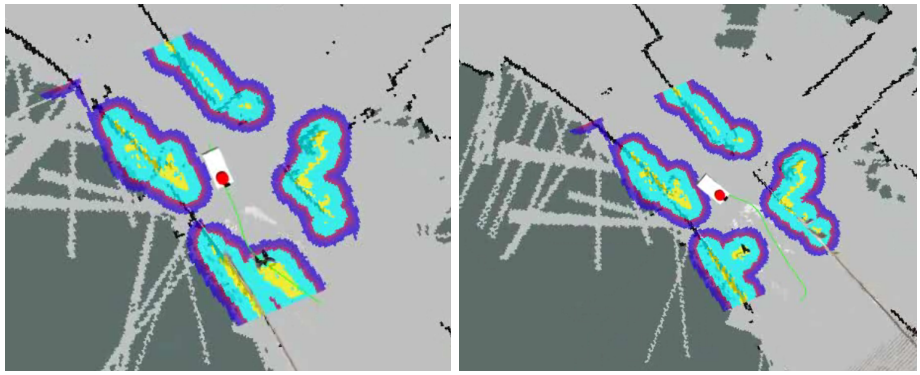


Figure 5.3: Nodes and topics for motion control.



(a) A person has moved into the path of the robot.

(b) A new path is planned, avoiding the new obstacle.

Figure 5.4: Moving obstacle avoidance. The local cost map, shown as coloured spots on the occupancy grid, is based on real-time sensor data.

Chapter 6
Discussion

6.1 Mapping

- Repairing broken maps? What to do when map is partially broken.
- Poor odometry when surrounding are in motion, or when laser features are difficult to detect. System can be fooled easily.

6.2 Navigation

- Rectangle base vs. square base.
- holonomic wheel.
- Open loop wheel control (stuck when friction is high.)
- Same as for mapping. Poor odometry when surrounding are in motion, or when laser features are difficult to detect. System can be fooled easily.

6.3 Suitability for Offshore Maintenance

This is just a prototype. Mobility issues. Kinect-like sensors and ROS could be useful. It is at least an excellent tool for ”rapid” prototyping.

6.3.1 Open Source Software and Security

ROS and other open source projects thrive active communities of contributors. Both Point Cloud Library (PCL) and ROS, as well as many other libraries and frameworks, are built on a collaborative effort from researchers and developers across the globe. While this is open structure is great for speeding up innovation. Issues and bugs can also be discovered more quickly by anyone. Another benefit is that every detail in an

open source project is open for scrutiny by those who want to use it. This is also a problem in terms of security. While anyone can find bugs and issues, the code is also open to those who are looking for possible exploits and vulnerabilities. If a system is targeted for sabotage, and it is widely known that the system uses open source software, it might be more vulnerable to security threats.

Chapter 7
Conclusion

7.1 Future Work

7.1.1 Autonomous Non-Destructive Testing

Advancements in Artificial Intelligence (AI), big data and machine learning opens up exciting possibilities for autonomous NDT. Branches of this technology is usually encountered in the context of image recognition, i.e. teaching machines to understand what they see. The same concepts may be applied to forms of NDT besides regular visual sensor input, such as ultra sound or eddy currents for corrosion detection.

7.1.2 Large Scale Kinect Fusion - Kintinous

Kinect Fusion has great potential for augmented reality. Augmented reality is a concept which blends the real and virtual environment. This opens up opportunities to create realistic and immersive training scenarios for the operators. Unfortunately, Kinect Fusion is limited reconstructing a rather small volume depending on the resolution. By varying the resolution, volumes can at the least cover a normal office desk and at the most cover a small room [**?**].

Kintinous...

A guide on how to build Kintinous can be found at https://github.com/mp3guy/Kintinuous. The procedure is complicated, as it usually is for experimental builds. It is recommended to attempt the procedure on a fresh install of Ubuntu 14.04 or 15.04 [Kin].

Improve the Communication Protocols

Communication between ROS and the XMEGA A3BU, the Bluetooth device and the OCS, all use the same pattern: A start byte `":"`, the message with the speed setting

and a stop byte `"Esc"`. In later projects, it could be beneficial to implement a more robust and rich communication protocol with more options for remote operation.

Implement a Fully Functional Operator Control Station

At the end of this project, the OCS provided functionality for moving the robot, and displaying live video from the Kinect.

7.1.3 Hardware

Several hardware-related issues became apparent over the course of the project - especially toward the final weeks. These issues are likely the results of many disconnected projects on the same hardware.

Kinect Sensor Location

This is the first semester in which a Kinect has been used on the robot. At the moment, the sensor is placed directly over the LIght Detection And Ranging (LIDAR) device. Because the depth sensor in the Kinect for XBOX 360 has a minimum range of roughly $0.5m$, it cannot detect objects within reach of the robot arm. It is recommended to find a new location further back on the robot.

Combine Stereo Cameras with Kinect-like Sensors

As mentioned, both active and passive depth cameras have limitations. The Kinect does function in direct sunlight, but it can measure depth in the dark. Passive depth sensors, for example stereo cameras, does depend on visible light to sense anything at all. While RTAB-Map does depend on visibility for loop closure detection, there are other SLAM methods, e.g. *Kinect Fusion*, which do not. An implementation could use a light sensor to sense light that may interfere with the Kinect. Light levels could be compared to a threshold and switch between the stereo cameras or the Kinect depending on how well each sensor will work in the current conditions.

On-board Computer Suitable for Moving Platforms

Because this author used his own computer to control the robot, all features related to ROS was removed from the robot at the end of the project. A new computer should be equipped with Solid State Drive (SSD) storage

Wheels

There were mainly two issues with the omni-wheels this semester: They are worn out, and one wheel slipped out of

the motor drive shaft. The rubber on a few of the perpendicular rollers is either loose or about to fall off the plastic rims. This causes the robot to shake, which can damage spinning hard disk drives or shake the sensors out of their calibrated positions.

7.2 Task Fulfilment

7.3 Final Conclusion

References

[ARG] ARGOS challenge. http://www.argos-challenge.com/en. Accessed: 19-05-2016.

[AS12] David A. Anisi and Charlotte Skourup. A step-wise approach to oil and gas robotics. In *Proceedings of the 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production*, June 2012.

[Asp13] Petter Aspunvik. Robotisert vedlikehold. Master's thesis, NTNU, 2013.

[ATE] ATEX directive: first edition of the atex 2014/34/eu guidelines. http://ec.europa.eu/growth/sectors/mechanical-engineering/atex/. Accessed: 13-05-2016.

[BMNK13] Kai Berger, Stephan Meister, Rahul Nair, and Daniel Kondermann. *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications: Dagstuhl 2012 Seminar on Time-of-Flight Imaging and GCPR 2013 Workshop on Imaging New Modalities*, chapter A State of the Art Report on Kinect Sensor Setups in Computer Vision, pages 257–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Bog16] Robert Bogue. Europe continues to lead the way in the collaborative robot business. *Industrial Robot: An International Journal*, 43(1):6–11, 2016.

[dep12] Sensabot: A safe and cost-effective inspection solution. *Journal of Petroleum Technology*, 64:32–34, 10 2012.

[DRC] DARPA robotics challenge. http://www.theroboticschallenge.org/overview. Accessed: 2016-04-05.

[ER12a] Mohamed A. El-Reedy. Chapter 6 - corrosion protection. In Mohamed A. El-Reedy, editor, *Offshore Structures*, pages 383 – 443. Gulf Professional Publishing, Boston, 2012.

[ER12b] Mohamed A. El-Reedy. Chapter 8 - risk-based inspection technique. In Mohamed A. El-Reedy, editor, *Offshore Structures*, pages 563 – 634. Gulf Professional Publishing, Boston, 2012.

[GC11] Peter Gorle and Andrew Clive. Positive impact of industrial robots on employment. Report, METRA MARTECH Limited, 2011.

[ifr] International federation of robotics - statistics. http://www.ifr.org/industrial-robots/statistics/. Accessed: 2016-04-05.

[JWA+12] J. Jamieson, L. Wilson, M. Arredondo, K. Evans, J.and Hamilton, and C Sotzing. Autonomous inspection vehicle: A new dimension in life of field operations. *Offshore Technology Conference*, April 2012.

[KFO12] Shinji Kawatsuma, Mineo Fukushima, and Takashi Okada. Emergency response by robots to fukushima daiichi accident: summary and lessons learned. *Industrial Robot: An International Journal*, 39(5):428–435, 2012.

[Kin] Kintinous. https://github.com/mp3guy/Kintinuous. Accessed: 2016-03-21.

[KMP15] K. Kydd, S. Macrez, and P. Pourcel. *Autonomous Robot for Gas and Oil Sites*. Society of Petroleum Engineers, September 2015.

[LM13] M. Labbe and F. Michaud. Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[LM14] M. Labbe and F. Michaud. Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666, Sept 2014.

[MIM] MIMROex, mobile maintenance and inspection robot for process plants. http://www.ipa.fraunhofer.de/fileadmin/user_upload/Kompetenzen/Roboter-_und_Assistenzsysteme/Industrielle_und_gewerbliche_Servicerobotik/English_Documents/Product_sheet_MIMROex_Mobile_maintenance_and_inspection_robot_for_process_plants.pdf.

[PBB11] K. Pfeiffer, M. Bengel, and A. Bubeck. Offshore robotics - survey, implementation, outlook. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 241–246, Sept 2011.

[pre] Press release: Subsea 7 completes design and build of first commercial autonomous inspection vehicle (aiv). http://www.subsea7.com/content/dam/subsea7/Company%20News/2011/Subsea7completesdesignandbuildoffirstcommericalAIV.pdf. Accessed: 2016-04-13.

[PSM+16] Ian Peerless, Adam Serblowski, Berry Mulder, et al. A robot that removes operators from extreme environments. In *SPE International Conference and Exhibition on Health, Safety, Security, Environment, and Social Responsibility*. Society of Petroleum Engineers, 2016.

[QGS15] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS*. O'Reilly Media, Inc., December 2015.

[RCR+15] Pere Ridao, Marc Carreras, David Ribas, Pedro J. Sanz, and Gabriel Oliver. Intervention auvs: The next challenge. *Annual Reviews in Control*, 40:227 – 241, 2015.

[Reb] Reboot ubuntu. http://askubuntu.com/questions/4408/what-should-i-do-when-ubuntu-freezes/36717#36717. Accessed: 2016-03-14.

[ROSa] ROS create package tutorial. http://wiki.ros.org/ROS/Tutorials/CreatingPackage. Accessed: 2016-02-05.

[ROSb] ROS history. http://www.ros.org/history/. Accessed: 2016-02-28.

[ROSc] ROS installation. http://wiki.ros.org/indigo/Installation/Ubuntu. Accessed: 2016-02-29.

[ROSd] ROS on the iss. http://www.ros.org/news/2014/09/ros-running-on-iss.html. Accessed: 2016-04-10.

[SWB+14] Samuel Soldan, Jochen Welle, Thomas Barz, Andreas Kroll, and Dirk Schulz. Towards autonomous robotic systems for remote gas leak detection and localization in industrial environments. In *Field and Service Robotics*, pages 233–247. Springer Berlin Heidelberg, 2014.

[Vin14] Jan-Erik Vinnem. *Offshore Risk Assessment vol 1.: Principles, Modelling and Applications of QRA Studies*. Springer London, London, 2014.

[WA12] Jarrett Webb and James Ashley. *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012.

Appendix A

Setting Up the Project

A.1 Installation

A.1.1 Equipment List

Item List

Software list

Hector SLAM for ROS Install with

```
sudo apt-get install ros-indigo-hector-slam
```

Compatibility Issues

Indigo, Ubuntu etc.

A.1.2 Install Ubuntu

A.1.3 Download ROS

A.2 Configuring the Project

A.2.1 Configuring the ROS Workspace

A.2.2 Configuring the Bluetooth Connection

The Qt framework is used to simplify the implementation of the Bluetooth connection between the ROS graph and a remote device. Our ROS installation for this project already includes some variant of Qt version 4.8. While useful for creating new GUI applications, it lacks a Bluetooth API. The latest version of Qt, version 5.x, is equipped with libraries necessary for developing Bluetooth applications. This part

of the guide will explain how to create a Qt 5 application which can be build by *catkin_make* and run as a *rosnode*.

1 - Install Qt5

Installing Qt5 for Linux is a straight forward procedure. Go to qt.io, and download the free version of Qt. All necessary instructions are provided. Qt5 may be installed in the home folder.

2- Enabling Qt5 in a ROS node

It is assumed that the ROS package ”bluetooth_server”, is located in a catkin workspace:

```
<NAME OF CATKIN WORKSPACE>/src/bluetooth_server
```

Inside this folder, open the file ”CMakeLists.txt” and locate the following:

```
set(CMAKE_PREFIX_PATH "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5"
  "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5Core"
  "/home/vegard/Qt/5.5/gcc_64/lib/cmake/Qt5Bluetooth"
```

Change these paths to the correct paths on your system.

Appendix B
Troubleshooting

B.1 Introduction

This chapter contains proposed solutions to some of the problems that was encountered over the course od the semester. The solutions are not complete or comprehensive, but may provide some quick fixes for any students that may continue working with this project.

B.2 ROS

B.2.1 ERROR: tf2:ExtrapolationException

When running the released ROS distribution binary of `RTAB-Map` installed with `apt-get` on the robot, the node would crash after a few iterations. The error message is as follows:

```
Lookup would require extrapolation into the future, ..., when looking
up transform from frame [laser] to frame [base_link].
```

The requested transform is milliseconds ahead of ”now”. This issue was fixed by maintainers in March 2016[1], but was not yet integrated into the released binary. During work with this project, the problem was solved by building `RTAB-Map` from source, where the most recent fixes are included. This is a straight forward procedure, which is described on the project's GitHub repository.

[1] https://github.com/introlab/rtabmap_ros/issues/54

B.3 Gazebo

B.3.1 Error [Node.cc.90] No namespace found

Solution: Remember to source the *gazebo* installation. In this case, with *gazebo-2.2* installed as recommended for ROS Indigo, the setup file can be sourced by typing

```
$ source /usr/share/gazebo-2.2/setup.sh
```

B.3.2 Dependency Issues When Installing *gazebo2*

This problem was encountered after removing gazebo and then typing

```
$ sudo apt-get upgrade
```

When typing

```
$ sudo apt-get install -y gazebo2
```

the installation failed because some dependencies had been upgraded to an incompatible version. To solve this, take note of the missing dependencies listed after entering the command above, open Ubuntu Software Center and select the History tab. Scroll down and locate the missing dependencies. They should have a red X next to them, indicating that they have been uninstalled. Then, enter the following command:

```
$ sudo apt-get install <NAME OF THE UNINSTALLED DEPENDENCY>
```

gazebo2

B.4 Ubuntu

B.4.1 Ubuntu Freezes

Sometimes during work with the project, Ubuntu would freeze and become unresponsive to keyboard input and mouse clicks. The mouse could be moved around, but was otherwise unresponsive. This event occurred exclusively when using *RVIZ* and displaying a camera topic as an image in the lower left corner of the GUI. The following steps from a post at askubuntu.com, solves the problem [Reb]:

While holding Alt and SysReq (Print Screen), type R E I S U B. Press each key properly, and allow a few seconds to pass between each keystroke so that each command has time to execute. This should cause the computer to reboot, and is supposedly safer than using the power button. Follow **this** link to the post for more details.